UNIVERSIDAD TECNOLÓGICA NACIONAL

Facultad Regional Buenos Aires

Sintaxis y Semántica de los Lenguajes

- 2024-

Docente: Dr. Oscar Bruno

Trabajo Integrador Primer Cuatrimestre

- ## Parte conceptual

Responda justificando con precisión su respuesta
1) Cuando se declara una variable de tipo puntero a cuantos espacios de memoria está asociado?
2) Para detectar el primer token de strtok debe invocarse con el string a analizar como primer parámetro
3) El complemento de una expresión regular es una expresión regular
4) Dada una lista de tipos, el tipo de una constante entera, determinado por su forma y valor, es el primero en el que puede ser representado
5) Complete con la definición formal del AFD requerido en la parte práctica

- ## Parte práctica

Escriba la expresión regular de las constantes enteras ([+,-] [0..9]) en C, con ello:

1. Obtenga por Semi Thompson el AF
2. Aplique el algoritmo de construcción de subconjuntos
3. Aplique el algoritmo de clases

- ## Complemento

Parte de C

1) Escriba un programa que determine e informe la cantidad de líneas que tiene un archivo de texto, que se pasa como argumento desde la línea de comandos. Debe resolverlo utilizando fgetc

Lenguajes formales y C

2) Escriba un programa comando que utilizando strtok y un autómata finito resuelva el siguiente problema “dado un archivo de texto con líneas de no más de 100 caracteres formadas por palabras separadas por blanco y dado además un carácter se debe crear otro archivo de texto con aquellas palabras en las que el carácter dado aparezca exactamente dos veces en la palabra. El archivo resultante debe tener una sola palabra por línea.

Parte teórica

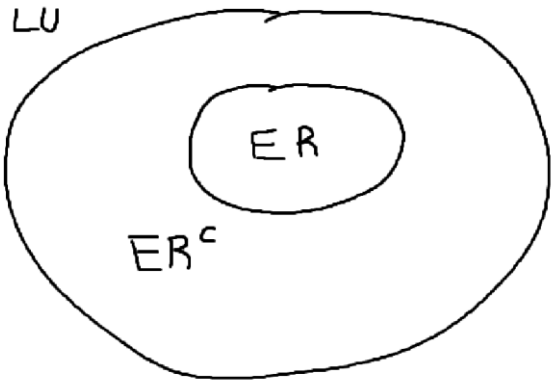
1) Los espacios de memoria a los que se puede asociar son dos: el primero, es el espacio de memoria donde se encuentra la dirección del puntero, mientras tanto, la segunda es el espacio de memoria a la que está apuntando.

   Son dos los espacios de memoria que asocia, la propia y a la que está apuntando.

2) **Verdadero**. Debido a que debe analizar el string completo para luego volver a usar strtok y separarlo por tokens.
   char* strtok (char*, const char*);
   En esta línea invoca el strtok para leer el string (corresponde a char*) y para recibir los delimitadores (corresponde const char*).

   Cuando llega al primer delimitador va a recibir todos los caracteres anteriores y para seguir usando el mismo puntero para seguir leyendo la cadena de caracteres entera habrá que invocarla devuelta con lo siguiente:
   char = (NULL, const char*).
   Con esta idea seguirá hasta el próximo delimitador y así hasta terminar.

3) Verdadero. Cuando hablamos del complemento de una expresión regular, aunque no tenga un operador oficial, nos referimos a todos los caracteres de un Lenguaje Universal el cual no era posible representar con esa expresión regular inicial.



4) En C, el tipo de una constante entera se determina en función de su valor y los sufijos opcionales, siguiendo un conjunto de reglas predefinidas por el estándar del lenguaje. Estas reglas aseguran que la constante se asigne al tipo más adecuado que pueda representarla sin pérdida de información y tenga menos desperdicio de espacio. Entre los tipos de datos podemos encontrar int, long int, unsigned long int.

Este ejercicio te pide que entiendas cómo se determina el tipo de una constante entera en un lenguaje de programación, basado en una lista de tipos predeterminada. La idea es que, cuando defines una constante entera en tu código, el compilador o intérprete decide qué tipo de dato utilizar para almacenar esa constante, y lo hace según un orden de tipos predefinido.

Ejemplo detallado:

1. **Lista de tipos predeterminada**:
    - Supongamos que tienes una lista de tipos en el siguiente orden: `int`, `long int`, `unsigned long int`.
    - Esto significa que el compilador primero intentará ver si el valor de la constante puede ser almacenado en un `int`. Si no puede, intentará con `long int`, y si no puede, intentará con `unsigned long int`.
2. **Constante entera**:
    - Cuando defines una constante entera en tu código, como por ejemplo `14`, el compilador decide cuál es el tipo más adecuado en la lista de tipos.
3. **Rango de `int`**:
    - Suponiendo que en el sistema que estás utilizando, un `int` se representa con 2 bytes (16 bits), el rango de valores que puede almacenar es de `-32768` a `32767`.
    - Esto significa que cualquier constante entera dentro de ese rango será del tipo `int`.
4. **Aplicación a las constantes**:
    - Si defines la constante `14`, como este valor cae dentro del rango de `-32768` a `32767`, se considera de tipo `int`.
    - Si defines la constante `-32769`, este valor está fuera del rango de un `int`. Por lo tanto, el compilador mirará el siguiente tipo en la lista, que es `long int`, y tratará de almacenarlo como tal.

Conclusión:

La regla principal es que el compilador asigna a la constante el primer tipo en la lista que puede representar el valor de la constante. Si el valor es muy grande o muy pequeño para el primer tipo en la lista, el compilador probará con el siguiente tipo, y así sucesivamente.

5) Un AFD es una tupla de 5 elementos, estos son:
    a) Q = conjunto de estados finito
    b) Σ = el alfabeto de caracteres que reconoce el autómata
    c) q0 = el estado inicial, esté si o si debe ser único
    d) f = el conjunto de estados finales.
    e) t = el resultado del conjunto de estados finitos y el alfabeto de caracteres.

Para ser AFD en la parte práctica debemos reconocer que no debe haber ninguna transición epsilon, siempre debe haber un único estado inicial y pueden existir varios finales.

Q = {0,1,2,3,4,5,6}

Sigma = {[0..9][A..F][a..f], Uu, Xx ε}

q0 = {0}

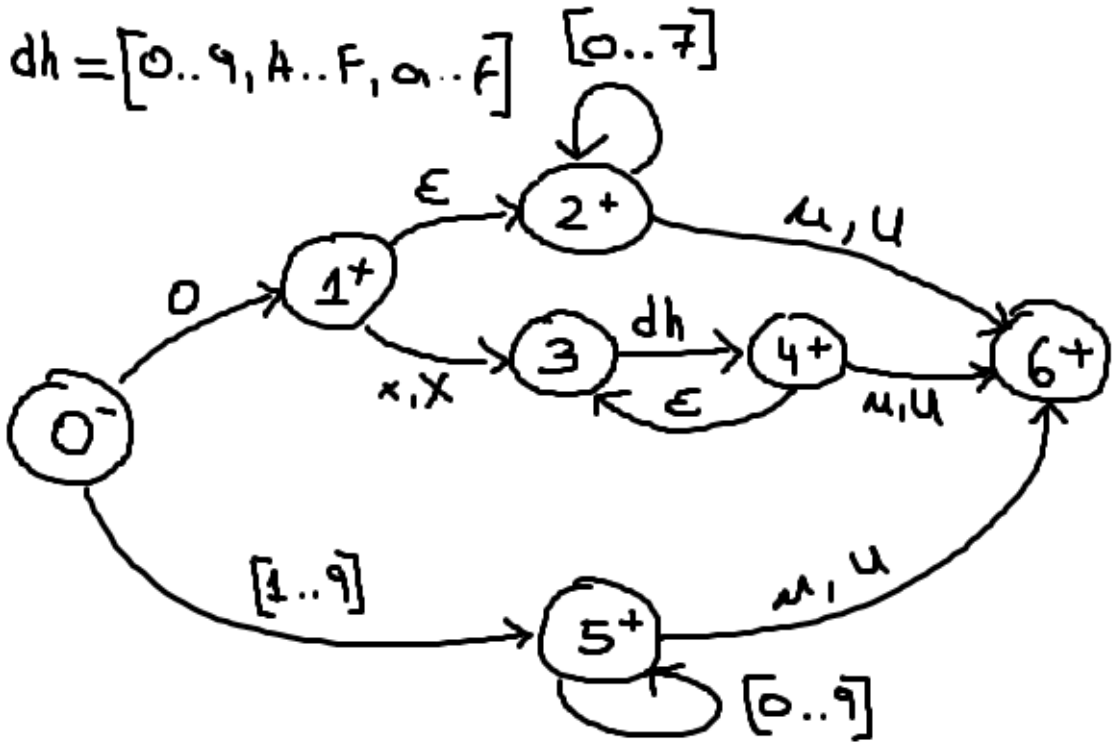
f = {1, 2, 4, 6, 5}

TT =

ESTADOS	0	[1..7]	[8,9]	[A..F, a .. f]	U,u	X,x
0-	1	5	5	7	7	7
1+	2	2	7	7	6	3
2+	2	2	7	7	6	7
3	4	4	4	4	7	7
4+	4	4	4	4	6	7
5+	5	5	5	7	6	7
6+	7	7	7	7	7	7
7	7	7	7	7	7	7

Parte Práctica

(0x+0X)+[0-9, A-F, a-f]?+(u+U)?+0[0-7]*(u+U)?+[1-9][0-9]*(u+U)?



ESTADOS	0	[1..7]	[8,9]	[A..F, a .. f]	U,u	X,x	ε
0-	1	5	5	-	-	-	-
1+	-	-	-	-	-	3	2
2+	2	2	-	-	6	-	-
3	4	4	4	4	-	-	-
4+	-	-	-	-	6	-	3
5+	5	5	5	-	6	-	-
6+	-	-	-	-	-	-	-

AFN -> AFD

clausura-epsilon(0) = { 0 }
hacia ({ 0 } , 0) = { 1 }
hacia ({ 0 } , [1..7]) = { 5 }
hacia ({ 5 } , [8,9]) = { 5 }

clausura-epsilon(5) = { 5 }
hacia ({ 5 } , 0) = { 5 }
hacia ({ 5 } , [1..7]) = { 5 }
hacia ({ 5 } , [8,9]) = { 5 }
hacia ({ 5 } , (u,U)) = { 6 }

clausura-epsilon(1) = { 1,2 }
hacia ({ 1,2 } , 0) = { 2 }
hacia ({ 1,2 } , [1..7]) = { 2 }
hacia ({ 1,2 } , (u,U)) = { 6 }
hacia ({ 1,2 } , (x,X)) = { 3 }

clausura-epsilon(6) = { 6 }
no hay hacia

clausura-epsilon(2) = { 2 }
hacia ({ 2 } , 0) = { 2 }
hacia ({ 2 } , [1..7]) = { 2 }
hacia ({ 2 } , (u,U) = { 6 }

clausura-epsilon(3) = { 3 }
hacia ({ 3 } , 0) = { 4 }
hacia ({ 3 } , [1..7]) = { 4 }
hacia ({ 3 } , [8,9]) = { 4 }
hacia ({ 3 } , [A..F, a..f]) = { 4 }

clausura-epsilon(4) = { 3, 4 }
hacia ({ 3,4 } , 0) = { 4 }
hacia ({ 3,4 } , [1..7]) = { 4 }
hacia ({ 3,4 } , [8,9]) = { 4 }
hacia ({ 3,4 } , [A..F, a..f]) = { 4 }

Hacemos las equivalencias para colocar en la tabla:

{ 0 } - = 0-
{ 1,2 } = 1
{ 2 }+ = 2+
{ 3 } = 3
{ 3,4 }+ = 4+
{ 5 }+ = 5+
{ 6 }+ = 6+

ESTADOS	0	[1..7]	[8,9]	[A..F, a .. f]	U,u	X,x
0-	1	5	5	-	-	-
1+	2	2	-	-	6	3
2+	2	2	-	-	6	-
3	4	4	4	4	-	-
4+	4	4	4	4	6	-
5+	5	5	5	-	6	-
6+	-	-	-	-	-	-

AFD -> AFD mínimo

Agrego el estado 7 para reemplazar los estados de rechazo vacíos

ESTADOS	0	[1..7]	[8,9]	[A..F, a .. f]	U,u	X,x
0-	1	5	5	7	7	7
1+	2	2	7	7	6	3
2+	2	2	7	7	6	7
3	4	4	4	4	7	7
4+	4	4	4	4	6	7
5+	5	5	5	7	6	7
6+	7	7	7	7	7	7
7	7	7	7	7	7	7

Ahora debo agrupar los estados finales y los no finales

ESTADOS	0	[1..7]	[8,9]	[A..F, a .. f]	U,u	X,x	Clases
0-	1	5	5	7	7	7	Clases C0
3	4	4	4	4	7	7	
7	7	7	7	7	7	7	
1+	2	2	7	7	6	3	Clases C1
2+	2	2	7	7	6	7	
4+	4	4	4	4	6	7	
5+	5	5	5	7	6	7	
6+	7	7	7	7	7	7	

No hay estados equivalentes dentro de esta tabla. Entonces realizo el siguiente paso.

Hacer la Tabla de Transiciones por Clases

ESTADOS	0	[1..7]	[8,9]	[A..F, a .. f]	U,u	X,x	Clases
0-	C1	C1	C1	C0	C0	C0	Clases C0
3	C1	C1	C1	C1	C0	C0	
7	C0	C0	C0	C0	C0	C0	
1+	C1	C1	C0	C0	C1	C0	Clases C1
2+	C1	C1	C0	C0	C1	C0	
4+	C1	C1	C1	C1	C1	C0	
5+	C1	C1	C1	C0	C0	C0	
6+	C0	C0	C0	C0	C0	C0	

Ahora debo agrupar las clases en sus equivalentes visibles en la tabla.

ESTADOS	0	[1..7]	[8,9]	[A..F, a .. f]	U,u	X,x	Clases
0-	C1	C1	C1	C0	C0	C0	Clase C0.0
3	C1	C1	C1	C1	C0	C0	Clase C0.3
7	C0	C0	C0	C0	C0	C0	Clase C0.7
1+	C1	C1	C0	C0	C1	C0	Clase C1.1
2+	C1	C1	C0	C0	C1	C0	
4+	C1	C1	C1	C1	C1	C0	Clase C1.4
5+	C1	C1	C1	C0	C0	C0	Clase C1.5
6+	C0	C0	C0	C0	C0	C0	Clase C1.6

Ahora, analizo la clase de C1.1 ya que tengo estados equivalentes por clases.

Es necesario descomponer la clase C1.1 para ver si son equivalentes

ESTADOS	0	[1..7]	[8,9]	[A..F, a .. f]	U,u	X,x
1+	2	2	7	7	6	3
2+	2	2	7	7	6	7

No se puede reducir debido a que los estados no son iguales.

Por lo tanto, la **tabla del AFD mínimo completo** es la siguiente:

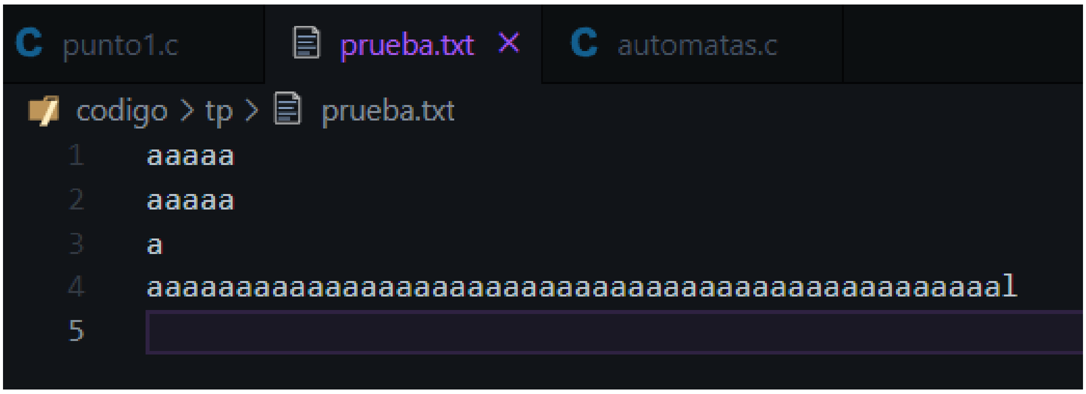
ESTADOS	0	[1..7]	[8,9]	[A..F, a .. f]	U,u	X,x
0-	1	5	5	7	7	7
1+	2	2	7	7	6	3
2+	2	2	7	7	6	7
3	4	4	4	4	7	7
4+	4	4	4	4	6	7
5+	5	5	5	7	6	7
6+	7	7	7	7	7	7
7	7	7	7	7	7	7

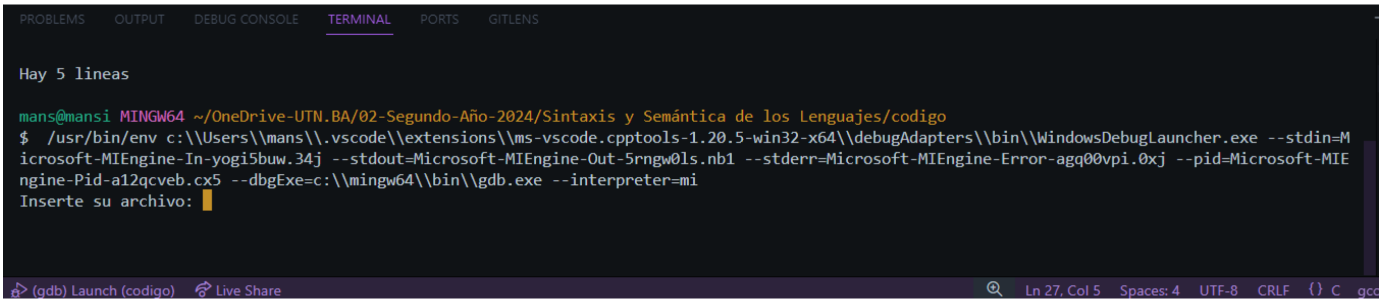
Se puede decir que el AFD original ya es mínimo.

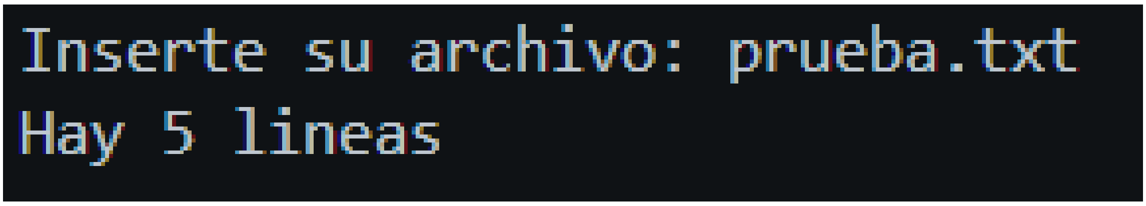
Parte C

Punto 1)

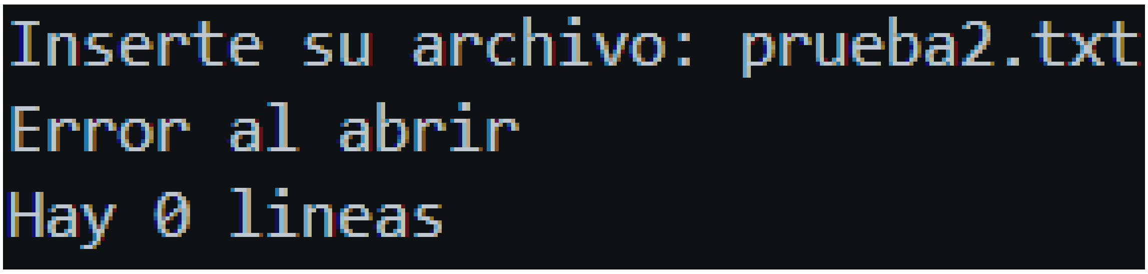
```c
#include <stdio.h>
int informar_lineas (char nombre[100], int *lineas);

int main(int argc, char** argv){
    int lineas = 1;
    char input[50];

    printf("Inserte su archivo: ");
    scanf("%50s",input);

    informar_lineas(input, &lineas);
    printf("Hay %d lineas\n", lineas);

    return 0;
}
int informar_lineas (char nombre[100], int *lineas){
    FILE *archivito = fopen (nombre, "r");
    fseek(archivito,0,SEEK_SET);

    if (archivito == NULL){
        fprintf (stderr, "Error al abrir\n");
        (*lineas)--;
        return -1;
}
    char aux;
    while((aux = fgetc(archivito)) != EOF){
        if (aux == '\n'){
            (*lineas)++;
        }
    }
    fclose(archivito);
    return 0;
```







En el caso de poner un archivo no válido saltaría…



Punto 2)

—---
SEMI-THOMPSON: un algoritmo para pasar de una ER a un AFND.
ALGORITMO DE CONSTRUCCIÓN DE CONJUNTOS O CLAUSURA EPSILON: pasamos del AFN al AFD
ALGORITMO DE CLASES: pasamos del AFD al AFD minimo.